The running time of a genetic algorithm or a random genetic algorithm depends mainly on the evaluation portion of each generation. Traditionally, each individual is evaluated by iterating through each of its genes and then deriving an overall fitness for that individual. Therefore, if there are n individuals in the population, and each consists of m genes, the time it takes to evaluate the population is O(n*m). Compounding on this is the fact that this is happens during each iteration, so this adds O(g*n*m) to the total running time of the program, where g is the number of generations.

To combat this compounding running time problem, the authors propose a combination of two techniques—distribution of the population with a divide and conquer method of the gene sequence.

The distribution for genetic algorithms can be accomplished by many different implementations (i.e. threads, network distribution), and traditionally serves as a platform by which the GA can evaluate its members in parallel. In the past, this has been done by dividing the population among the available parallel evaluators. In our model, however, we propose not dividing the population, but instead dividing the Chromatin.

Using the divide and conquer methodology as is common to improve search algorithm time, we consider the chromosomes to be the search space. If there is at least one set of chromosomes that will produce a predetermined maximum fitness value, then we can reduce the size of the search space by dividing the Chromatin. Each evaluator then works on a population which is a subset of the overall solution. In basic trials, the results of dividing the individual were that each evaluator was able to reach a maximum fitness in a very short number of generations because the subset of the genes on which it worked was trivially solved via combinatorics to evolve to the maximum fitness level.

This is opposed to what might be considered an enumerative method that eventually considers all sets of possibilities and tried to solve the problem. This is a rather expensive way to go about it considering if we know an individual to be y characters with x being the number of possible characters expressed. The enumerative method could essentially consider all (x!)^y possibilities that would lead to the fittest individual, even in distribution models the evaluation of so many individuals would take a rather long time. In our method consider a number z that is an integer divisor of y, this leads to (x!)^z being considered by evaluation node in the system by creating a population that is less then or equal to (x!)^z, call this population size j, even in the most exhaustive runs it'd take j * g where g was the number of generations. j * g is by construction significantly smaller then (x!)^y. This being distributed through the system solves the chromosomal problem much faster then the problem of the individuals.

The solution of these sets of trivial problems leads to applying the Bolzano-Weierstrass theorem that at its core says "the convergence of the set of all subsequences implies the convergence of the sequence". This tells us that if an individual's chromosomes are all evaluated as the fittest possible, the individual must therefore be the fittest individual; all of this at a much lower computational cost then enumerative methods.